Design of a PC-based Open Industrial Robot Control System Integrated with Real-Time Machine Vision*

Yanan Liu[1], Tianliang Hu[2], Hepeng Ni[2], Chengrui Zhang[2], Bao Lin[2] and Euan Judd[1]

Abstract— **This paper presents a new industrial robot control system integrated with real-time image processing based on a Windows PC. This enables both machine vision and robot control tasks to be carried out on a single general-purpose computer rather than using two separate systems to perform different tasks. To accomplish this, a three-layer software structure is developed along with a series of standard peripheral devices in the hardware layout. Real-time performance of the system is tested and we demonstrate the system on a DELTA parallel manipulator. Experimental results show that the manipulator can pick up 120 objects on a moving conveyor per minute. This work demonstrates the feasibility of integrating real-time machine vision tasks with robot control tasks without dedicated hardware.**

I. INTRODUCTION

In recent years, PC-based robot control systems and machine vision systems have been widely used for different manufacturing processes. Robotics and machine vision have both undergone rapid changes with advanced systems continually making existing products obsolete [1].

Industrial robots are an important part of high-end manufacturing equipment, with high technology and wide application range [2]. However, the booming market for industrial automation led to a vendor-dominated market. Currently, the majority of industrial robot control system vendors adopt their own structure with specified programming languages, a variety of operating systems and non-standard hardware interfaces [3], which results in separate control systems, a large number of incompatible products, frequent updates for controllers, high after-sale service costs and a dependence on professional personnel. This situation limits the reconfigurability of motion control systems and hinders the development of robot control systems, which has led to the demand for open-architecture controllers.

To deal with the incompatibility problem that arises from various vendors and realise high-speed industrial applications which require images to be captured and processed within a specific time, a PC-based robot control system integrated with real-time machine vision is needed. In this paper, the vision system utilises a general-purpose PC as the controller and standardised Network cards as framegrabber cards which supports a standard interface: GigE Vision [4]. With this method, both good computational performance and utilisation of low-priced standard hardware can be achieved.

With the improvement of PC hardware performance, its CPU can fulfill the demands as the core components of a numerical system. PCs have been used as a standard platform for open architecture control systems [5]. The use of PCs is expected to increase rapidly and the main reasons for the widespread PC-based controls is the emergence of control network standards for motion and the I/Os that allow easier integration of various peripheral devices. Also, easy modification and extension of operating systems is another important factor for the PC-based control [6]. Thus, the facility to integrate other functionalities (e.g. machine vision) is a strong reason to use standard PC hardware in robot control open architectures [7].

In this work, we propose a robot control system that adopts a PC + Windows OS + Kithara Real-time Suite(KRTX) [8] mode in which all the robot and machine vision algorithms are executed in the Windows OS on one PC without requiring any other processing hardware. With this system, real-time machine vision can be realised in a robot controller with high flexibility.

II. THE ARCHITECTURE OF KRTS

To integrate robot control system and real-time machine vision on a PC, the software platform should be well designed. This paper adopts KRTS as the real-time extension suite for Windows OS. The KRTS is developed by Kithara Software GmbH which can be regarded as a real-time OS extension that requires no modifications to the standard Windows OS kernel and needs limited modifications to the standard Windows OS hardware abstraction layer (HAL). Its application programming interface (API) can be called to expand an OS into two OSs running in parallel. One of them is the non-real-time OS (non-RTOS, Windows OS) with rich software and hardware resources, the other is a real-time subsystem, Kithara RTS-Kernel, with hard real-time performance. Compared to the real-time transformation of traditional non-RTOS, KRTS does not need to cut or configure the system kernel but adds a Kithara RTS-HAL. Fig. 1 shows the KRTS architecture and its main features.

A number of methods for communication between the communication layer and real-time layer are available, including shared memory, pipes and mailslots. EtherCAT Master function is a feature of KRTS which supports other fieldbus protocols such as Profibus, CANopen, EtherMAC [9] as well. KRTS also supports the acquisition of images

*This work was supported by National Natural Science Foundation of China (Grant No. 51405270), China Scholarship Council (No. 201700260083), and the Bristol Robotics Laboratory.

[1]Yanan Liu is with the Bristol Robotics Laboratory, University of Bristol, Bristol, BS16 1QY, United Kingdom (e-mail: yanan.liu@bristol.ac.uk)

[2]Chengrui Zhang is with the School of Mechanical Engineering, Shandong University and Key Laboratory of High Efficiency and Clean Mechanical Manufacture, Ministry of Education, Shandong University, Jinan, 250061, China (corresponding author, phone: +86-0531-88392700-1; e-mail: crzhang@sdu.edu.cn)

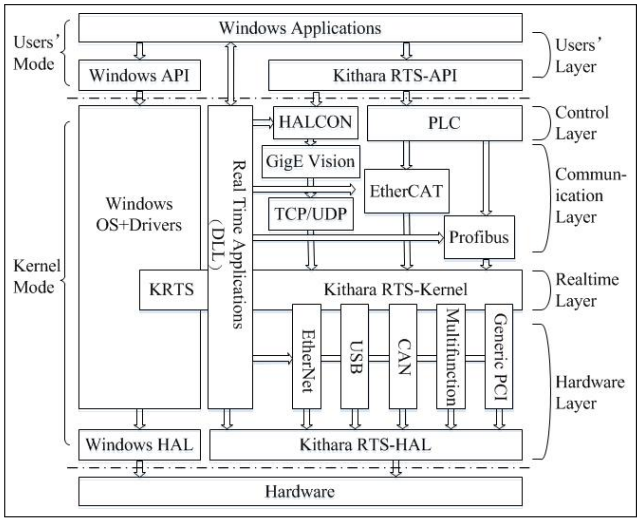

Fig. 1. The Architecture of KRTS [8].

from GigE Vision cameras and processing with OpenCV and HALCON in real-time.

III. System Hardware Configuration

Our proposed system hardware structure consists of controllers, communication units, I/O units, data memory, user interface devices and sensors (Fig. 2). In this structure, the controller adopts a PC running Windows 7 OS and extends three network ports for connection with a teach pendant, a motion control adapter and an industrial camera respectively. The information data exchanged between a controller and external devices (e.g. cameras) is transferred through standard Ethernet cards. For communication with servo motor drivers, a coupler or gateway supports various communication protocols such as EtherCAT, EtherMAC, CANopen, Profibus. Other standard features include Transmission Control Protocol/Internet Protocol (TCP/IP) which permits both communication with local devices (e.g. handheld units and cameras) and external communications with standard and customer applications.

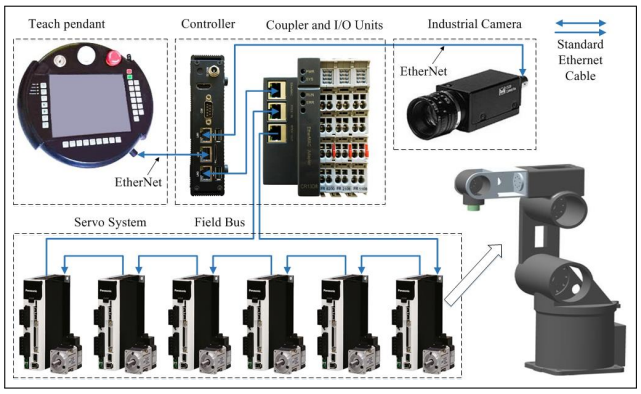

Fig. 2. Hardware topology of the robot control system.

As a teaching tool, the teach pendant is an important part of the industrial robot control. The teach pendant is a handheld device which is used with a robot controller to move, program, and run industrial robots. With a teach pendant, users can set robot parameters, view position information, check running status, and read alarm messages. The machine vision system can be integrated directly into the robot controller. Industrial cameras connect to an Ethernet port in the controller through GigE Vision that is a widely-adopted interface. There are dozens of leading companies currently offering hundreds of GigE Vision compliant products. The benefits of GigE Vision are the interchangeability of hardware and the industrial approach.

IV. System Software Design

To meet the needs of the real-time machine vision and open architecture control system, a modular design is required (Fig. 3) based on the hardware configuration. This system is split into three layers which consist of different modules with specific functions.

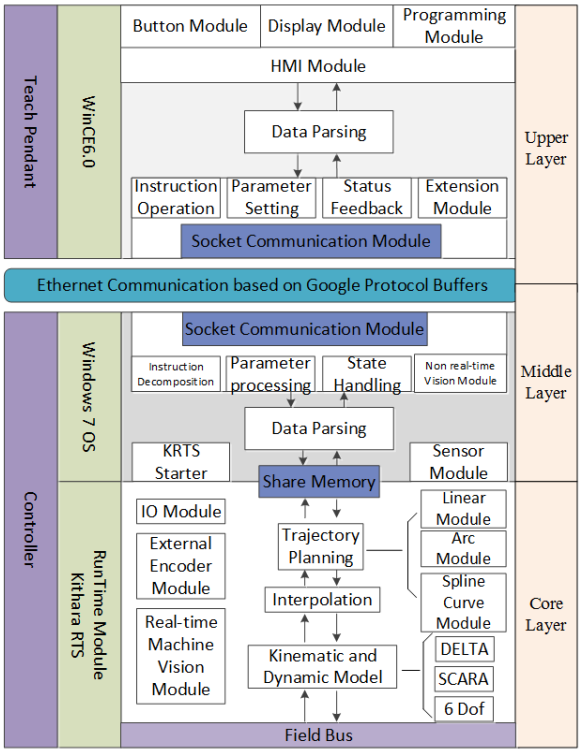

Fig. 3. Software structure of the robot control system.

A. Upper Layer

The upper layer focuses on design of the teach pendant software based on an embedded OS Windows CE 6.0. Teach pendant is an interface between an operator and a robot. During an interaction process, operators carry out a series of operation and management tasks through the teach pendant, such as menu switching, parameter setting and code editing. An embedded computer is the main controller with a human-machine interface designed using Microsoft's open-source API. This layer has two tasks, human-computer interaction and Ethernet communication.

As shown in Fig. 3, the HMI Module consists of a Button Module, a Display Module, and a Programming Module. The Button Module sends functional commands to the robot control system by using a touch screen or buttons. The Display Module displays robot motion and system information including axis angles, end-effector positions, system status, and I/O status. The Programming Module is used to teach programming and execute programs. When users operate the teach pendant, information from the HMI Module is parsed, after that, lower-layer functions (e.g. Instruction Operation Function, Parameter Setting Function) deals with those instructions. Meanwhile, the Status Feedback Function receives information from the middle layer, then sends this information to the HMI module.

The teach pendant adopts a WinCE OS which is modular with the strong communication capabilities of an embedded OS. The WinCE OS is specifically designed to develop various portable personal computing devices; thus, it is suitable for the development of teach pendants. The communication between the teach pendant and controller requires a reliable and fast response speed [10]. In the implementation of communication between the teach pendant and controller, a Socket Communication Module is developed based on Google Protobuf [11], which sends instructions from the upper layer to the middle layer with a standard Ethernet interface. Protobuf simplifies the packing and parsing of the transmission and, therefore, it is easy to transfer a great deal of information, including images, with a high transmission speed. In conclusion, the use of Windows CE OS simplifies the development process as well as obtaining a fast response time and good extensibility. With standard interfaces and existing APIs, using an embedded computer as the teach pendant both simplifies and shortens the development period.

B. Middle Layer

The middle layer is the interface between the upper layer and the core layer and is mainly responsible for shared memory and for establishing communication between the upper layer, core layer and middle layer. Its functionality also include launching KRTS, loading real-time kernel dynamic link library (DLL) and transferring data bidirectionally.

The Vision module runs in this layer (Non Real-time Vision Module) or the core layer (Real-time Machine Vision Module) according to different machine vision requirements. The Vision Module employs a vast number of well-known image processing methods. OpenCV programming functionality is used for both real time and non-real time applications. Also, other commercial image processing software and hardware can be supported as long as they are adaptable to a Windows OS.

C. Core Layer

All real-time tasks are encapsulated into a DLL and loaded into the core layer by KRTS API. After starting KRTS, two of the cores in a multi-core computer are allocated for real-time motion control and real-time vision. In this way, the performance of the system is improved, and execution time is parallelised in different cores. This layer uses kinematic and dynamic models, a trajectory planning module, a interpolation module, an I/O module and a communication module to perform real-time tasks. This system is open to a variety of robots like DELTA, Scara and other 6 Degree-of-Freedom (DOF) robots if the kinematics and dynamics are properly established. In particular, the core layer supports real-time machine vision and provides a good platform for visual servoing [12].

V. Real-Time Performance Analysis

A real-time system is a computer system capable of performing computations or processing transactions and responding to external events within a certain time. Real-time performance is a very important part of industrial robot control [13]. Currently, there are many exclusively real-time operating systems available Table I. We choose the Windows OS with real-time extension considering Windows supports nearly all the software and hardware in the market, indicates better generality and portability compared to other schemes. In addition, the development environment and development tools in the Windows environment is rich.

TABLE I
Different types of RTOS.

RTOS types	Product examples
Independent structure	VxWorks, QNX, WinCE, usOS-II, INTERGRITY, LynxOS
Real-time expansion based on Linux	RTAI, Xenomai, RTcore, Monta Visa Linux, TimeSys Linux, Red-Linux
Independent structure based on Windows	KRTS, InTime, RTX, Hyperkernel

This paper adopts the interrupt service latency and the timer jitter as the real-time performance index [14]. These two parameters are related to the computer hardware configuration that is shown in Table II.

TABLE II
Configuration of the motion control system and machine vision system.

Item	Model
CPU	Intel Atom E3825 1.33GHz
RAM	DDR3 SDRAM 2GB
Hard Disk	BIWIN 6202 32GB
Network Chip	Realtek RTL8111F 10/100/1000 Mbs
Hardware Platform	Lex 2I260D single board machine
Operating System	Windows7 Ultimate
Development Environment	Visual Studio 2010
Camera	Basler acA640-120gm
Camera resolution	659px * 494px
Camera Frame Rate	120 fps
Camera Interface	GigE
Sensor Type	CCD

A. Controller Response Time Distribution

The computer's responses to external interface devices and performing the real-time periodic task scheduling is realised by the hardware interrupt. As a result, interrupt service

latency is one of the most basic parameters of computer real-time performance [15]. In this motion control system, the interrupt service latency refers to the response time of the real-time subsystem to the interrupt signal of Ethernet cards, that is, the duration between the controller network card receiving the data packets and the controller starting to parse the data packets. However, in practice, it is difficult to measure the duration directly. For this system, we use a router to monitor the data packets sent to the PC and received from the PC. Wireshark software is applied to record the sending packet time Ts and the time the packet is received Tr on another monitoring computer (Fig. 4).

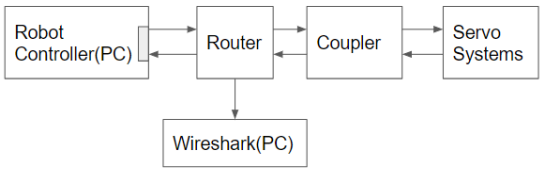

Fig. 4. The response time measurement system with a router and Wireshark software.

The controller response time can be represented

$$\Delta t = (Tr - Ts)/2 \tag{1}$$

In this system, the data packets contain the position control information.

The core-layer real-time cycle period for the motion control system is set to 1 ms and runs for 4 h continuously. While the motion control system is running, the latency time Δt is recorded. N refers to number of corresponding delay times recorded. As we can see from Fig. 4, the response times of this system are distributed between 11 and 35 μs. The largest latency time is 35.43 μs which indicates that this controller meets the real-time control needs.

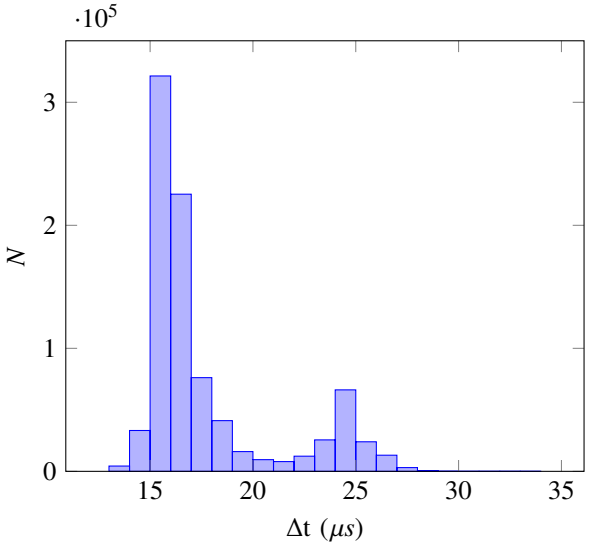

Fig. 5. Controller response time distribution.

B. Kithara Timer Jitter Test

Periodic real-time tasks are the most common forms of application in real-time systems and they are normally realised by a timer. KRTS provides high-frequency timers for Windows which is used for robot control and image processing in this paper. Periodic tasks enter the execution state at regular intervals of time. Timer jitters indicate the accuracy of a timer, so it is an important index to evaluate the real-time performance of a system. In this test, we use the Kithara Performance Analyzer [8] to measure the Kithara timer jitter. The period $T = 1$ ms and resolution $R = 1$ μs is recorded for 10 h. As Fig. 6 illustrates, the jitter distributes between 1 and 6 μs and the worst-case timer jitter is less than 6 μs which verifies the accuracy and stability of a Kithara timer.

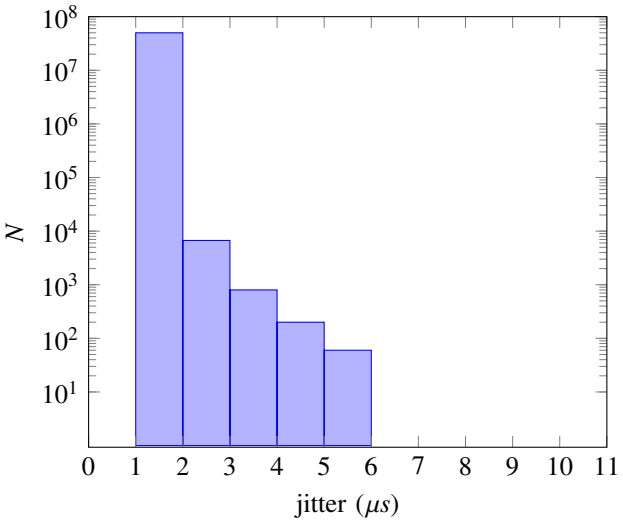

Fig. 6. Kithara timer jitter distribution.

C. Real-time vision test

The real-time vision performance of KRTS is tested in order to determine whether the acquisition and processing times are deterministic. The test program demonstrates the execution of an OpenCV3.0 application in real-time in conjunction with the acquisition of images from a GigE Vision camera. The program loads a DLL on the kernel-level, which contains image processing algorithms and algorithms for handling image reception. The duration between image acquisition and image processing is recorded for 45 consecutive minutes. Images are obtained by KRTS and then processed by OpenCV in real-time. Image processing algorithms are used to get the binary images from the source images, and then find circles using circular Hough transform (Fig. 7, Fig. 8).

As we can see from Fig. 9, the image acquisition and image processing time distribution is between 12.1 and 13 ms and therefore has stable and reliable real-time performance with this hardware configuration and image processing algorithm.

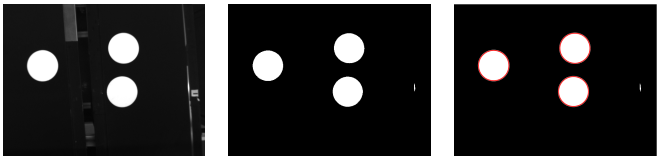

Fig. 7. Find circles from source images. left: source image, middle: binary image, right: detected circles.

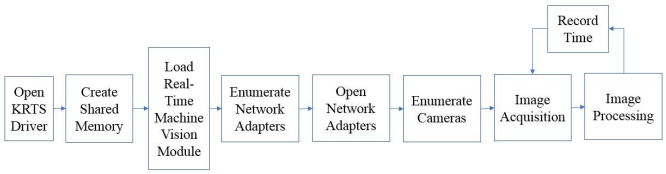

Fig. 8. Real-time image capturing and processing flowchart.

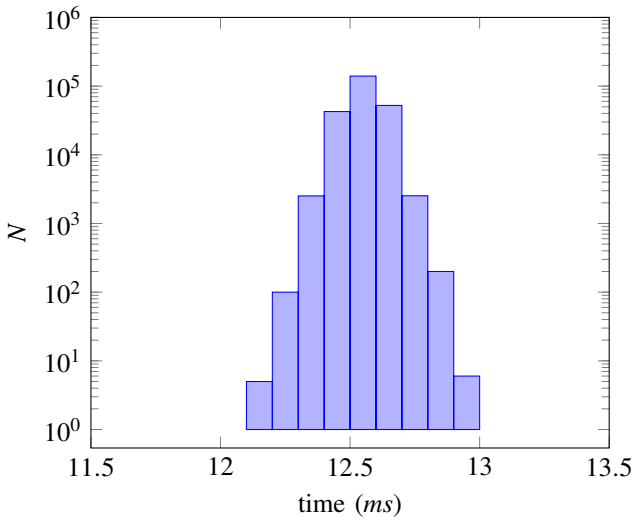

Fig. 9. Image acquisition and processing time distribution.

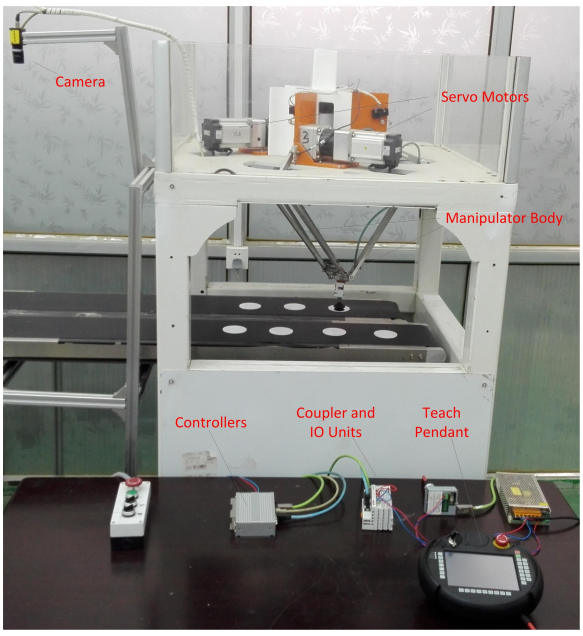

Fig. 10. The DELTA manipulator sorting system.

VI. Prototype Validation

A. A DELTA manipulator sorting system

Based on the robot control platform illustrated in this paper, a DELTA manipulator sorting system (Fig. 10) was developed. The main purpose of DELTA manipulator sorting system is to carry out high-speed manipulations in different applications [16]. Fig. 10 depicts the system configuration that includes a camera, two conveyors, a DELTA manipulator, a controller, a teach pendant, a coupler and I/O units.

During operation, the camera captures images of objects on a conveyor and sends these images to the controller through a Gigabit Ethernet card. The vision module then processes the images and the position of an object is obtained. The position information of the object, obtained from the global variables, is then received by the runtime module. After that, according to the object position and conveyor velocity, the grabbing position is predicted. Then, the controller plans the trajectory and the DELTA manipulator moves along the trajectory to meet the objects on the moving conveyor belt. The detailed procedure is described in Fig. 11.

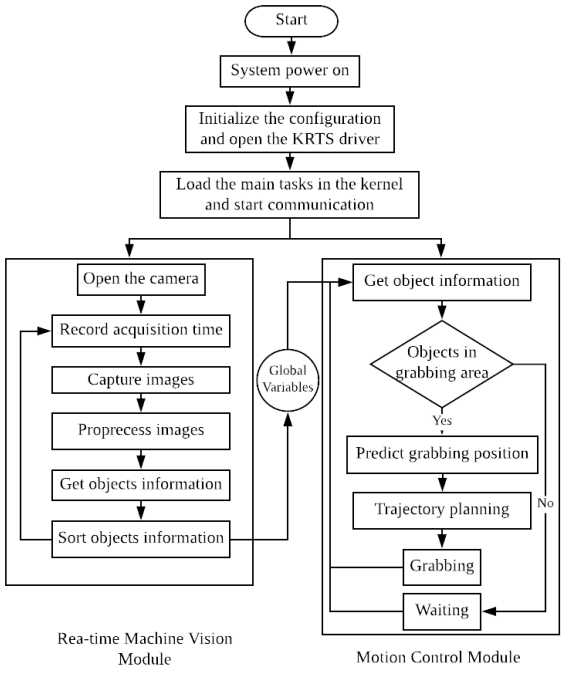

Fig. 11. The Operation process of motion control system.

The platform's stability and sorting performance are tested when it is running under different conveyor velocities over a 8 *h* duration with a 1 *ms* communication period. The test results are seen in Table III in which *Vcon*, *Nsum*, *Nloss*, *Nerr* represent the conveyor velocity, total number of picked objects, number of missed objects, and the number of mistakenly picked objects respectively.

It is clear from Table III that both the platform stability and sorting performance are accurate for all *Vcon* which indicates the open control system integrated real-time machine vision

is reasonable. Moreover, the DELTA manipulator picking frequency can reach 120 times per minute with an average horizontal distance between the picking point and placing point 30 *cm*, and a largest end effector lifting distance of 20 *cm*. Despite this, the picking frequency was not sufficient to pick up all objects on the first pass if the conveyor belt was too densely packed, in which case *Nloss* would be greater than zero. However, all objects would be successfully picked up given enough passes of the the conveyor belt.

TABLE III
GRABBING RESULTS OF DELTA MANIPULATOR SORTING SYSTEM FOR THE FIRST CONVEYOR BELT PASS.

Vcon(mm/s)	Nsum	Nloss	Nerr
100	6723	1	0
150	6530	0	0
200	6179	1	0

VII. CONCLUSION

This work demonstrated an industrial robot control system based on a general purpose PC for integrating real-time machine vision and robot motion control. This reduces incompatibility between the vision system and robot motion control system. We analysed the controller performance and proved that it had real-time ability with the largest response latency of 35.43 μs with the above-mentioned system configuration. The controller gives fast and stable vision processing. Our trials suggest that the vision subsystem can achieve real-time circle detection at over 75 FPS.

The combination of the real-time machine vision and robot technology plays an important role in the field of industrial automation. As a high speed robot application, the DELTA sorting system has been shown to be capable of picking up moving objects rapidly based on the proposed robot controller. For future applications, the versatility and real-time features of this platform are ideal for visual servoing systems and systems requiring multisensory fusion.

REFERENCES

[1] A. Pugh, *Robot vision*. Springer Science & Business Media, 2013.

[2] T.-m. Wang and Y. Tao, "Research status and industrialization development strategy of chinese industrial robot," *Journal of mechanical engineering*, vol. 50, no. 9, pp. 1–13, 2014.

[3] A. Bonarini, M. Matteucci, M. Migliavacca, and D. Rizzi, "R2p: An open source hardware and software modular approach to robot prototyping," *Robotics and Autonomous Systems*, vol. 62, no. 7, pp. 1073–1084, 2014.

[4] W. He, K. Yuan, H. Xiao, and Z. Xu, "A high speed robot vision system with gige vision extension," in *Mechatronics and Automation (ICMA), 2011 International Conference on*, pp. 452–457, IEEE, 2011.

[5] M. Tooley, *PC based instrumentation and control*. Taylor & Francis, 2013.

[6] K.-S. Hong, K.-H. Choi, J.-G. Kim, and S. Lee, "A pc-based open robot control system: Pc-orc," *Robotics and Computer-Integrated Manufacturing*, vol. 17, no. 4, pp. 355–365, 2001.

[7] A. S. de Oliveira, E. R. De Pieri, D. Martins, and U. F. Moreno, "A five-layers open-architecture robot controller applied to interaction tasks," in *Robotic Symposium, 2008. LARS'08. IEEE Latin American*, pp. 184–189, IEEE, 2008.

[8] KitharaSoftware, "Real-time for windows." `http://kithara.com/en`. (Date last accessed 18-March-2018).

[9] K. Wang, C. Zhang, X. Ding, S. Ji, and T. Hu, "A new real-time ethernet for numeric control," in *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*, pp. 4137–4141, IEEE, 2010.

[10] N. Wang, F. Zhang, and X. Zhang, "Development of industrial robot teach pendant based on wince," in *Mechanism and Machine Science*, pp. 249–260, Springer, 2017.

[11] J. Feng and J. Li, "Google protocol buffers research and application in online game," in *Conference Anthology, IEEE*, pp. 1–4, IEEE, 2013.

[12] B. Jia, S. Liu, K. Zhang, and J. Chen, "Survey on robot visual servo control: Vision system and control strategies," *Acta Automatica Sinica*, vol. 41, no. 5, pp. 861–873, 2015.

[13] L. Madan, K. Anand, and B. Bhushan, "Real-time operating system," *International Journal of Research in Science And Technology*, no. 4, pp. 39–50, 2014.

[14] D. Honegger, H. Oleynikova, and M. Pollefeys, "Real-time and low latency embedded computer vision hardware based on a combination of fpga and mobile cpu," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 4930–4935, IEEE, 2014.

[15] P. Hambarde, R. Varma, and S. Jha, "The survey of real time operating system: Rtos," in *Electronic Systems, Signal Processing and Computing Technologies (ICESC), 2014 International Conference on*, pp. 34–39, IEEE, 2014.

[16] S. B. Park, H. S. Kim, C. Song, and K. Kim, "Dynamics modeling of a delta-type parallel robot (isr 2013)," in *Robotics (ISR), 2013 44th International Symposium on*, pp. 1–5, IEEE, 2013.